Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matija Andrejčič

Analiza vpliva tipa čakalnih vrst na zakasnitve paketov

DIPLOMSKO DELO
UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

prof. dr. Miha Mraz
MENTOR

Ljubljana, 2015

Tematika naloge:

Kandidat naj v svojem delu izvede analizo vpliva strežne discipline na zakasnitev paketov v računalniških omrežjih. Pri tem naj se usmeri na FIFO, prioritetno in uteženo krožno strežno disciplino. Analizo naj podkrepi s simulacijskimi rezultati uporabe navedenih strežnih disciplin na M/M/1/s strežnem sistemu.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Matija Andrejčič izjavljam, da sem avtor pričujočega diplomskega dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih, kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelal samostojno pod mentorstvom prof. dr. Mihe Mraza,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike dela v zbirki "Dela FRI".

— Matija Andrejčič, Ljubljana, avgust 2015.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matija Andrejčič

Analiza vpliva tipa čakalnih vrst na zakasnitve paketov

POVZETEK

V diplomski nalogi smo opredelili probleme zagotavljanja kakovosti storitve interaktivnih aplikacij, ki zahtevajo prenos podatkov v realnem času. Predstavili smo osnovne pojme strežbe ter se seznanili s problemom čakalne vrste. Z vidika teorije strežbe smo predstavili dva primera realnih izgubnih strežnih sistemov s končno čakalno vrsto. Problem čakalnih vrst rešujemo z razvrščevalnimi algoritmi. Poznamo več vrst razvrščevalnih algoritmov, ki jih lahko med seboj primerjamo na podlagi različnih meril. Izbrane algoritme smo uporabili na zgledu simulacije sistema M/M/1/s in jih podrobneje analizirali. Eksperimente smo izvedli v programskem okolju OMNeT++. Slednje nam omogoča postavitev simulacijskega modela, izvedbo simulacije in analizo pridobljenih rezultatov. Rezultate simulacij smo uporabili za analizo in primerjavo učinkovitosti izbranih razvrščevalnih algoritmov.

Ključne besede: kakovost storitve, čakalna vrsta, razvrščevalni algoritem, FIFO algoritem, algoritem s prioriteto, algoritem z obteženim krožnim razvrščanjem, OMNeT++

University of Ljubljana
Faculty of Computer and Information Science

Matija Andrejčič

Analysis of the influence of queue type on packet delay in computer networks

ABSTRACT

In this thesis, we have identified the problems of ensuring the service quality of interactive applications, which require data transfer in real time. We realised basic concepts of network service and got acquainted with the problem of waiting queue. In view of network service theory, we introduced two examples of real loss service systems with final waiting queue. The problem of waiting queue is solved with scheduling algorithms. There are several kinds of scheduling algorithms, which can be compared by different criteria. Selected algorithms have been used in simulation of system M/M/1/s and analysed in more detail. The experiment was done in software environment OMNet++, which makes it possible to set up a simulation model, carry out the simulation and analyse the results. The results of the simulation have been used to analyse and compare the efficiency of the selected scheduling algorithms.

Key words: quality of service, waiting queue, scheduling algorithm, FIFO algorithm, strict priority algorithm, weighted round robin algorithm, OMNeT++

ZAHVALA

Zahvaljujem se mentorju prof. dr. Mihi Mrazu za strokovno pomoč pri izdelavi diplomske naloge. Zahvala gre tudi vsem prijateljem, ki so mi v času študija stali ob strani. Posebej bi se rad zahvalil staršema, ki sta mi študij omogočila in me podpirala skozi študijsko pot.

— Matija Andrejčič, Ljubljana, avgust 2015.

KAZALO

1 Uvod

V zadnjem desetletju smo bili priča hitremu razvoju računalniških omrežij in njihovih storitev. Posledično so se pojavile potrebe po večji zmogljivosti in obvladovanju prometa, ki se prenaša po omrežju. Vse več je prisotnih interaktivnih aplikacij, ki zahtevajo prenos podatkov v realnem času. Primer takšnih aplikacij sta video konferenca in IP telefonija. Zaradi nizke cene sta vse bolj priljubljeni med končnimi uporabniki. Takšne aplikacije zahtevajo prenos podatkov v vnaprej določenem času, ki nam dopušča dovolj časa za rekonstrukcijo signala. Kot končni uporabniki, si ne želimo motenj pri pogovoru s prijateljem, kaj šele izgube vsebine. Zgornja meja izgube paketov, ki je še sprejemljiva pri navedenih aplikacijah, znaša okoli 2.5% [1]. Drugi problem predstavljajo zakasnitve. Eden izmed mehanizmov za zmanjšanje zakasnitev je uvedba prioritete posamezne vrste prometa in uporaba učinkovitih razvrščevalnih algoritmov, s čimer pripomoremo k boljši kakovosti storitve (angl. *Quality of Service*).

V pričujočem delu se bomo najprej seznanili z osnovnimi pojmi strežbe v računalniških omrežjih. Za komunikacijo sta potrebna vsaj dva računalnika, kjer eden deluje kot strežnik, drugi pa ima vlogo odjemalca. S povečevanjem števila uporabnikov, se po-

sledično povečuje tudi promet v omrežju in zahteve po boljši kakovosti storitve. V nadaljevanju bomo spoznali dva tipa storitev [2], in sicer *integrirane* ter *diferencirane* storitve. Spoznali bomo naprave, ki so danes prisotne v računalniškem omrežju. Pred napravo se običajno nahaja vhodna čakalna vrsta, ki hrani zahteve preden gredo v strežbo. Na koncu drugega poglavja se bomo srečali s problemom čakalnih vrst in spoznali vzroke za zakasnitve in izgube paketov. Zakasnitve paketov so odvisne od več dejavnikov. Med njih štejemo zasedenost omrežja, število napak pri prenosu, zmogljivost in rezervacija omrežnih virov ter zasedenost čakalnih vrst. V diplomskem delu se bomo osredotočili na vhodne čakalne vrste, katere upravljajo različni razvrščevalni algoritmi. V tretjem poglavju bomo z vidika teorije strežbe spoznali dva primera realnih izgubnih strežnih sistemov, in sicer M/M/1/s ter M/M/m/m. V praktičnem delu bomo simulirali strežni sistem M/M/1/s, ki vsebuje končno vhodno čakalno vrsto. Za upravljanje čakalne vrste skrbi razvrščevalni algoritem, ki določa, kateri izmed paketov v vhodni čakalni vrsti bo poslan v strežbo. Na koncu tretjega poglavja bomo spoznali različne razvrščevalne algoritme in merila na podlagi katerih jih lahko med seboj primerjamo. Preden se bomo lotili praktičnega dela diplomske naloge, si bomo ogledali in analizirali tri obstoječe simulacije različnih razvrščevalnih algoritmov. V zaključku diplomskega dela se bomo seznanili s programskim orodjem OMNeT++, v katerem bomo implemetirali tri različne razvrščevalne algoritme končne vhodne čakalne vrste strežnega sistema M/M/1/s. Opisali bomo konfiguracijo modela omrežja in za vsak razvrščevalni algoritem izvedli tri ponovitve simulacije. Za analizo bomo uporabili povprečne rezultate simulacij. Pridobljene podatke bomo grafično predstavili in na njihovi podlagi algoritme med seboj primerjali.

2 Opis problema

V pričujočem poglavju bomo spoznali osnovne pojme strežbe, ki jih srečamo v računalniških omrežjih. Osredotočili se bomo na problem zagotavljanja kakovosti storitev končnemu uporabniku, izkoriščenost omrežnih naprav, na koncu pa bomo spoznali problematiko čakalnih vrst.

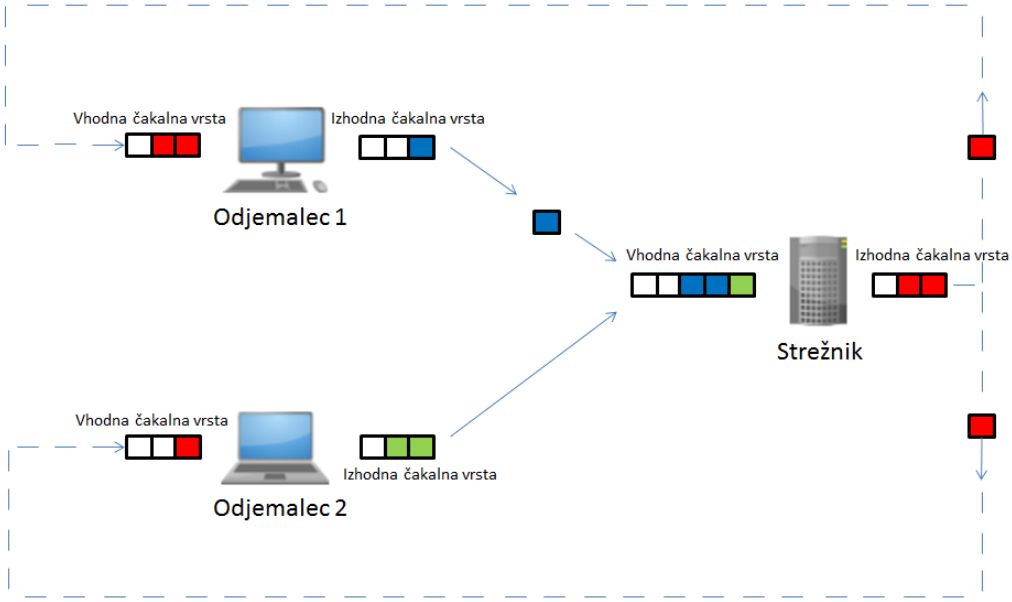
2.1 Osnovni pojmi strežbe paketov v računalniških omrežjih

Hiter razvoj računalniških omrežij posledično povzroča potrebo po novih načinih upravljanja z veliko količino prometa. Prenos podatkov preko omrežja danes poteka po delih, ki so enkapsulirani v pakete. Najpogosteje se uporabljata protokola TCP in UDP. TCP nam zagotavlja pravilen prenos paketov preko omrežja s potrjevanjem. UDP nam tega ne omogoča, ker gre za nepovezovalni protokol, kjer odjemalec pakete pošilja brez preverjanja, ali so se paketi uspešno prenesli. UDP paketi so temu primerno manjši in se uporabljajo, kjer je potrebna velika hitrost in izguba paketov nima večjega vpliva na zagotavljanje kvalitete storitve. Za komunikacijo sta potrebna vsaj dva računalnika, ki delujeta po principu odjemalec/strežnik [3]. Primer delovanja takšnega sistema je pri-

kazan na sliki 2.1. Odjemalca pošiljata zahteve strežniku, po končani strežbi pa jima lahko ta glede na tip zahteve vrne odgovor. V primeru, da je strežnik zaseden, gredo paketi v čakalno vrsto. Čakalna vrsta je majhen medpomnilnik, namenjen shranjevanju paketov preden gredo v strežbo. Poznamo dva tipa čakalnih vrst, in sicer vhodne in izhodne [4]. Vhodne so namenjene shranjevanju paketov, preden gredo v strežbo. Izhodna čakalna vrsta vsebuje pakete pripravljene za pošiljanje v omrežje. Večina aplikacij deluje na način, da na začetku pošiljajo pakete z najvišjo hitrostjo. Čakalne vrste hitro postanejo polne, zato prihaja do zakasnitev in izgube paketov. Pri TCP se v primeru izgube paketa le-ta pošlje ponovno, kar predstavlja dodaten promet v že preobremenjenih vrstah in na prenosnih poteh [5]. TCP protokol v primeru, da pošiljatelj ne prejme potrditve, čas pošiljanja povečuje z zamikom dvojne, štirikratne, osemkratne,... dolžine začetnega intervala, dokler ne doseže primerne hitrosti pošiljanja paketov. Pri UDP protokolu, kjer se potrjevanja ne uporablja, se paketi preprosto zavržejo brez ponovnega pošiljanja. Velikost izhodnih čakalnih vrst se običajno meri v številu paketov in jo lahko ponekod spreminjamo, če imamo na voljo dovolj pomnilnika. V operacijskem sistemu Linux imamo na voljo v ta namen posebne ukaze. Primer takšnega ukaza je `ifconfig eth0 txqueuelen 1000`. S tem ukazom nastavimo velikost izhodne čakalne vrste vmestnika eth0 na 1000 paketov. Paketi so lahko različnih vrst, prioritet in velikosti. Vse to vpliva na čas strežbe, zato je potrebno učinkovito upravljanje čakalnih vrst, ki skrbi za dobro izkoriščenost omrežnih virov in hkrati zagotovi kakovost storitev za končnega uporabnika.

2.2 Kakovost storitev

Hitra rast uporabe interneta in razvoj novih storitev sta povečala tudi potrebo po zagotavljanju prenosa podatkov v realnem času. Tukaj se srečamo s problemom, kako zagotoviti kakovost storitve (angl. *Quality of Service*) za nemoteno delovanje aplikacij. Pojem kakovosti storitev je zelo širok. Ena izmed splošnih definicij, objavljena v [6], je sledeča: "Kakovost storitve je skupni učinek lastnosti storitve, ki določajo zadovoljstvo uporabnika." Ta problem je prisoten predvsem pri interaktivnih aplikacijah kot so videokonference in IP telefonija, kjer je potreben prenos slike in zvoka v realnem času. Tu je potrebno omeniti, da se za običajen prenos videa uporablja predpomnilnik, kamor se pred predvajanjem naloži del podatkov, ki se nato predvajajo. Prednost takega načina je, da nam omogoča večjo toleranco zakasnitev ali izpada povezave, saj imamo na pred-



Slika 2.1 Na sliki sta prikazana dva odjemalca in strežnik, puščice pa prikazujejo smer prenosa paketov.

vajalniku dovolj podatkov za začasno nemoteno predvajanje vsebine. Problem se pojavi pri interaktivnih aplikacijah, kjer gre za interakcije v realnem času in si tega ne moremo privoščiti, saj bi bile zakasnitve prevelike. Take aplikacije zato zahtevajo dostavo paketov v vnaprej določenem času, ki dopušča aplikaciji rekonstrukcijo in predvajanje vsebine. Glede občutljivosti zakasnitev delimo aplikacije v dve skupini [7]:

- Elastične aplikacije: zakasnitve paketov nimajo velikega vpliva na obnašanje aplikacij; sem spadajo aplikacije za prenos datotek in elektronskih sporočil;
- Toge aplikacije: zakasnitve imajo velik vpliv na obnašanje aplikacij; primer takšne aplikacije je prenos zvoka in videa v realnem času.

Zaradi nizke cene se vse pogosteje uporablja za prenos zvoka IP omrežje. Ta način prenosa je zelo občutljiv na zakasnitve, ki mora biti po priporočilu mednarodne zveze za telekomunikacije ITU-T manjša od 150ms. Kot končni uporabniki si želimo tekočo komunikacijo oziroma komunikacijo brez izgubljenih podatkov in zakasnitev. Za dobro delovanje aplikacij želimo imeti izgube paketov manjše od 1%, zgornja še sprejemljiva meja pa je 2.5%. Izgube višje od 10% so že nedopustne in zelo moteče za končnega upo-

rabnika [1]. Promet skozi omrežje se stalno spreminja, z njim pa tudi polnjenje čakalnih vrst. To povzroča spreminjanje zakasnitev, ki ga imenujemo trepetanje signala (angl. *jitter*). Zakasnitve so sestavljene iz fiksnega dela, ki je odvisen od hitrosti povezav in razdalje fizične poti med pošiljateljem in sprejemnikom. Temu času se doda spremenljivi del, ki je odvisen od čakalnih časov paketov v čakalnih vrstah na poti med pošiljateljem in prejemnikom. Na fiksne zakasnitve ne moremo vplivati, medtem ko bi z ustrezno disciplino čakanja lahko zagotovili boljši pretok paketov z višjo prioriteto. Nekateri paketi, za katere zakasnitve niso kritične, so lahko postreženi kasneje in s tem omogočimo strežbo paketov z višjo prioriteto. Parametri, ki vplivajo na kakovost storitve, so sledeči [6]:

- pasovna širina,
- frekvenca zgostitev prometa,
- verjetnost in število napak pri prenosu,
- rezervacija omrežnih virov,
- zakasnitve in nihanje zakasnitev,
- zasedenost omrežja,
- fizična razdalja,
- varnost in zasebnost podatkov in
- zasedenost čakalnih vrst.

S strani IETF organizacije za standardizacijo internetnih protokolov je bila predlagana delitev storitev [2] na integrirane ter diferencirane.

2.2.1 Integrirane storitve

Integrirane storitve (angl. *Integrated Services*) temeljijo na rezervaciji poti med izvorom in ponorom, za kar se uporablja protokol RSVP. Aplikacija, ki potrebuje visoko kvaliteto storitve, mora rezervirati pot med izvorom in ponorom. Takšen način delovanja bi spremenil nepovezavno omrežje v povezavno orientirano omrežje [2]. Slabost tega pristopa je, da morajo vsi usmerjevalniki na poti podpirati protokol RSVP, s čimer se poveča kompleksnost implemetacije usmerjevalnika. Vsak usmerjavalnik si mora zapomniti sejo med izvorom in ponorom, poleg tega pa mora imeti strategijo razvrščanja sej odvisno od QoS parametrov. Integrirane storitve delimo na 3 vrste [8]:

- "Best effort" - storitev najboljši možni način, ki velja za današnji internet;
- "Controlled-Load Service" - storitev z nadzorovano obremenitvijo; uporablja se v primerih ko aplikacija potrebuje večjo pasovno širino, kot storitev "best effort"; kakovost storitve je približno enaka, kot pri neobremenjenem omrežju;
- "Guaranteed Service" - zagotovljena kakovost storitve; namenjena je aplikacijam s strogo omejenimi zakasnitvami in prepustnostjo.

Slabost integriranih storitev je, da je pri vsakem spreminjanju usmerjevalne tabele potrebno ponovno nastaviti rezervacije. Zahteva večjo kompleksnost umerjevalnika, ki mora znati klasificirati in razvrščati pakete. Potrebna je politika kontrole, kdo lahko izvede rezervacijo. Takšen način ima nizko skalabilnost zaradi ogromnega števila sej. V ta namen je bila razvita alternativna rešitev diferenciranih storitev.

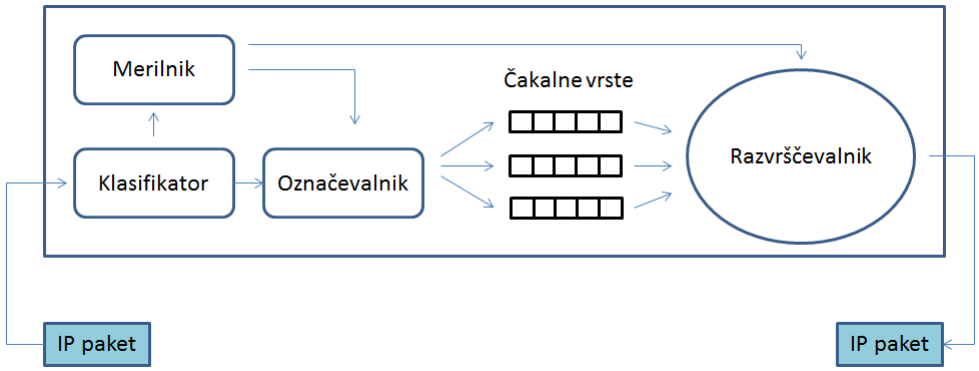
2.2.2 Diferencirane storiteve

Diferencirane storitve (angl. *Differentiated Services*) [9] ne potrebujejo shranjevanja sej v usmerjevalnikih. Delujejo po principu prometnih razredov, v katere se razvrščajo pretoki paketov glede na vnaprej določena pravila. Za klasifikacijo se uporablja polje ToS (angl. *Type of Service*), ki se nahaja v glavi IP paketa. Polje ToS je sestavljeno iz 8 bitov, kjer prvi trije predstavljajo prednostne bite. Biti za klasifikacijo v prioritetne razrede so predstavljeni v tabeli 2.1. Usmerjevalniki znotraj omrežja razporejajo pakete v različne čakalne vrste glede na polje ToS. Promet razvrstijo glede na prioritetni razred. Vsak usmerjevalnik mora biti konfiguriran v skladu s prioritetami. Na sliki 2.2 je prikazana logična arhitektura usmerjevalnika. Sestavljajo jo naslednji elementi:

- Klasifikator: razvrsti pakete v interne razrede, glede na razred, ki mu ga določi usmerjevalnik na robu omrežja;
- Merilnik: meri lastnosti tokov paketov, ki so izbrani s strani klasifikatorja; Daje navodila označevalniku in razvrščevalniku za upravljanje vsakega paketa v skladu z vnaprej določenimi pravili;
- Označevalnik: označi paket glede na razred določen s strani klasifikatorja;
- Razvrščevalnik: pakete iz čakalnih vrst glede na vnaprej določena pravila pošilja v omrežje.

Prednostni biti	Prioriteta
111	Network Control
110	Internetwork Control
101	Express Forwarding
100	Razred 4
011	Razred 3
010	Razred 2
001	Razred 1
000	Best effort

Tabela 2.1 Tabela prikazuje prednostne bite, ki se uporabljajo za klasifikacijo razredov.



Slika 2.2 Prikaz logične arhitekture usmerjevalnika [9].

V primerjavi z integriranimi storitvami so diferencirane bolj skalabilne in manj procesno potratne, ker ni potrebno shranjevanje in obdelovanje sej v usmerjavalnikih. Popolne zagotovitve kakovosti storitve ni, saj omrežni viri niso rezervirani, kot pri integriranih storitvah. V praksi se diferencirane storitve uporabljajo predvsem v hrbteničnih omrežjih, integrirane pa v lokalnih omrežjih pri kritičnih aplikacijah za zagotavljanje kakovosti storitve.

2.2.3 RSVP protokol

Protokol RSVP (angl. *Resource Reservation Protocol*) [10] deluje na transportni plasti in je podrobneje opisan v RFC 2205 [11]. Uporablja se za rezervacijo virov med pošiljateljem

in sprejemnikom za zagotavljanje kakovosti storitve. Deluje v IPv4 in IPv6 omrežjih. Omogoča rezervacijo virov za unicast in multicast promet. Tok paketov je določen s ciljnim IP naslovom, identifikatorjem protokola in ciljnimi vrati. Vsi usmerjevalniki na poti morajo podpirati protokol RSVP. Za rezervacijo se uporabljata dve vrsti sporočil:

- Sporočilo PATH: shranjuje pot do sprejemnika; v vsakem vozlišču se shranjuje IP naslov prejšnjega vozlišča, ki je vsebovan v PATH STATE; v primeru, da sporočilo pride do usmerjevalnika, ki ne podpira RSVP protokola, sporočilo posreduje naprej brez rezervacije vira;
- Sporočilo RESV: pošlje ga sprejemnik pošiljatelju nazaj po isti poti, s čimer rezervira vire v primeru, da so ti na voljo.

2.3 Izkoriščenost strežnikov in ostalih omrežnih naprav

V računalniškem omrežju danes srečamo veliko število naprav, ki s časom še vedno narašča. Med te naprave sodijo:

- strežniki,
- odjemalci,
- fizični prenosni mediji,
- usmerjevalniki,
- stikala in ostale naprave.

Vsaka izmed njih ima določeno nalogo, ki jo mora opravljati. Naloga usmerjevalnika je usmerjanje prometa skozi omrežje po najboljši možni poti glede na ciljni IP naslov. Deluje na omrežni plasti OSI modela, medtem ko stikalo deluje na transportni plasti in usmerja pakete glede na strojni MAC naslov znotraj lokalnega omrežja. Strežniki ponujajo različne storitve, odjemalci generirajo zahteve in tako dalje. Na strežnikih lahko teče več aplikacij, ki so med seboj običajno neodvisne z namenom, da izpad ene ne vpliva na delovanje ostalih. Delujejo po principu odjemalec-strežnik, kjer odjemalec pošlje zahtevo, strežnik pa sprocesira odgovor. Strežnik lahko teče na domačem računalniku, običajno pa se uporablja namenski računalnik s specifično strojno in programsko opremo. Takšen računalnik vsebuje večje število procesorjev, več delovnega pomnilnika, mrežnih

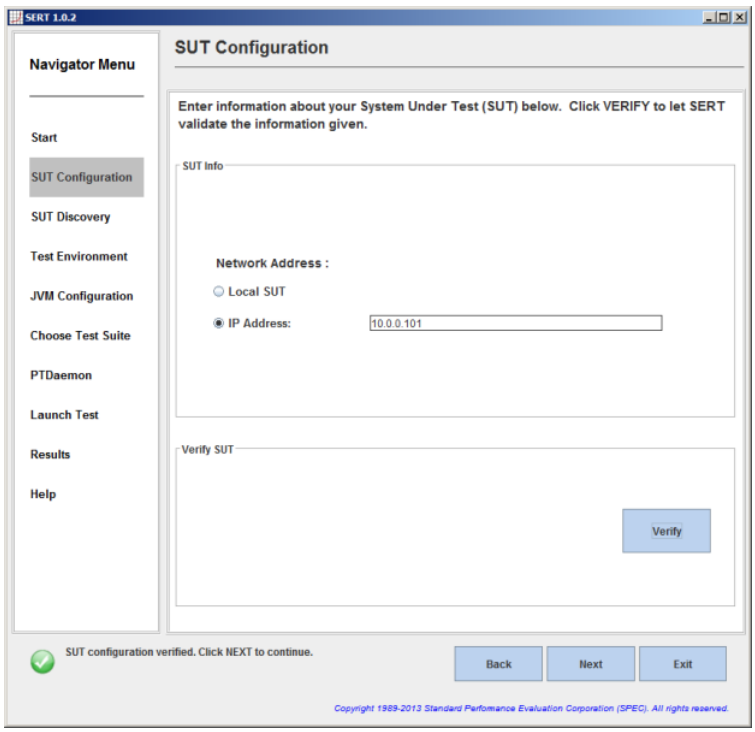
kartic in ostalih komponent. Vse to nam omogoča potrebno zmogljivost za normalno delovanje spletnih storitev. Poleg storitev, ki jih strežnik ponuja, mora zagotoviti tudi kontrolo dostopa in varnost. Glede na storitve poznamo več vrst strežnikov [12]:

- Aplikacijski strežnik: na njem tečejo določene aplikacije do katerih lahko dostopamo preko brskalnika;
- Poštni strežnik: upravlja prenos in dostop do elektronskih sporočil;
- Podatkovni strežnik: običajno vsebuje bazo podatkov in sistem za upravljanje z bazo podatkov;
- Datotečni strežnik: omogoča shranjevanje in dostop do datotek;
- Igralni strežnik: uporablja se za povezovanje odjemalcev računalniških iger za igranje preko interneta;
- DNS strežnik: omogoča preslikovanje imen v IP naslove;
- Proxy strežnik: deluje kot posrednik za anonimno brskanje po spletu;
- Strežnik za tiskanje: upravlja dostop do tiskalnika;
- Spletni strežnik: namenjen za gostovanje spletnih strani.

Nakup strežnika je relativno draga zadeva. Poleg nakupa strojne in programske opreme je potrebno neprekinjeno napajanje, vzdrževanje ter redundanca, s katero preprečimo izpad delovanja in izgubo podatkov. Končnim uporabnikom želimo zagotoviti hiter in nemoten dostop. Za dobro izkoriščenost strežnika je potrebno učinkovito upravljanje s paketi v vhodni čakalni vrsti, za kar potrebujemo algoritem za razvrščanje paketov.

2.4 Orodje SERT

Merjenje zmogljivosti strežnika lahko opravimo z orodjem SERT (angl. *Server Efficiency Rating Tool*), razvitim s strani organizacije SPEC (angl. *Standard Performance Evaluation Corporation*) [13], ki je bila ustanovljena leta 1988. SPEC je postal eden izmed vodilnih konzorcijev za meritev zmogljivosti. Na sliki 2.3 je prikazan grafični vmesnik orodja SERT.



Slika 2.3 Grafični vmestnik orodja SERT [14], kjer nastavimo IP naslov sistema za testiranje.

Orodje za testiranje uporablja sintetično breme, s katerim skušamo posnemati realno breme pri najmanjši, povprečni in največji obremenitvi sistema. Primeri bremen so sledeči [15]:

- CPE: množenje matrik s plavajočo vejico, kompresija, enkripcija podatkov;
- Omrežni V/I: branje in pisanje paketov, prepustnost, latenca;
- Pomnilniški V/I: dostop do baze podatkov, branje in pisanje datotek, prepustnost diska.

Z njim testiramo različne komponentne kot so procesor, delovni pomnilnik, zunanji pomnilnik ter porabo energije pri različnih obremenitvah. Rezultat testiranj je prikazan v človeku razumljivi obliki, primerni za nadaljno obdelavo podatkov. Z analizo testiranj lahko odkrijemo in odpravimo napake sistema, ki bi nas v kasnejših fazah razvoja drago stale.

2.5 Problematika čakalnih vrst

V omrežjih se pretakajo velike količine podatkov, ki se z razvojem stalno povečujejo. S povečanjem prometa se poraja potreba po zmogljivejših omrežnih virih. Ker so omrežni viri dragi, pogosto niso dovolj zmogljivi za sprotno obdelavo paketov, zato potrebujemo medpomnilnik, ki mu pravimo čakalna vrsta. Paket, ki pride v čakalno vrsto, mora počakati, dokler se strežnik ne sprosti. Pred tem je potrebno obdelati pakete, ki so bili že v čakalni vrsti. V omrežjih so zato pogosto prisotne zakasnitve, ki jih je potrebno odpraviti z ustreznimi mehanizmi. Eden izmed njih je strežna disciplina. Njena naloga je razvrščanje paketov v čakalnih vrstah. Najpreprostejša disciplina, ki se uporablja v večini sistemov, je FIFO (angl. *First In First Out*), ki jo bomo kasneje tudi natančneje opisali. V tem primeru so paketi postreženi v istem vrstnem redu, kot so prispeli. Problem se pojavi pri velikem toku paketov, ki povzročijo zakasnitve, lahko pa tudi izgube paketov. Razlogi za izgube paketov so lahko različni:

- premajhna čakalna vrsta, zaradi napačnih nastavitev ali premajhnega pomnilnika;
- premajhna zmogljivost omrežnih virov; do te napake najpogosteje pride zaradi slabega načrtovanja omrežja; zmogljivost sistema je potrebno testirati pri različnih obremenitvah in z analizo odpraviti napake;
- preobremenitev strežnika v izjemnih primerih; eden iz med njih se je zgodil pri prodaji vstopnic za enega izmed največjih festivalov Ultra Europe 2014, kjer je bil izjemen naval nakupa vstopnic, ki je po nekaj minutah povzročil zrušitev strežnika [16];
- internetni napadi DoS (angl. *Denial of Service*) so podrobneje opisani v [17]; napad se izvede preko množice računalnikov, ki pošljejo veliko število zahtev, s katerimi preobremenijo in onemogočijo storitev strežnika; primer napada se je zgodil na WordPress, kjer je bilo poslanih več Gbitov/s oziroma 10 miljonov paketov na sekundo, s čimer so onesposobili 18 miljonov blogov [18];
- kvaliteta storitve, kjer so kritične aplikacije močno odvisne od zakasnitev in zahtevajo, da so njeni paketi postreženi v določenem času; s tem izpodrivajo pakete z nižjo prioriteto, ki so v primeru polne čakalne vrste lahko tudi zavrženi, odvisno od algoritma za razvrščanje čakalnih vrst; primera kritičnih aplikacij sta IP telefonija in IP televizija.

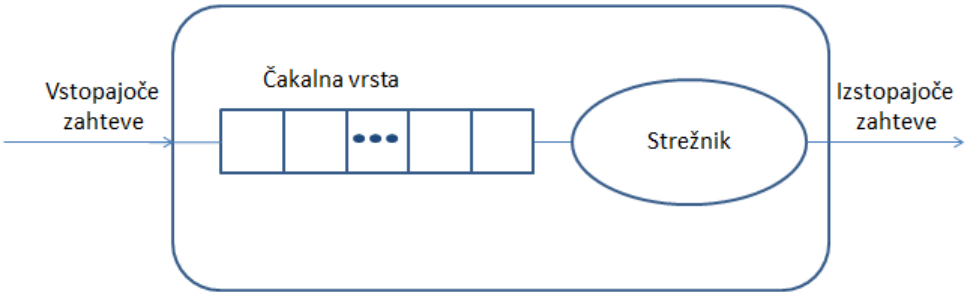
3 Pristopi k reševanju problema

V pričujočem poglavju bomo spoznali osnovne pojme teorije strežbe, kjer bomo predstavili dva primera strežnih sistemov. Sledi predstavitev algoritmov za razvrščanje paketov v čakalnih vrstah in opis meril za oceno razvrščevalnih algoritmov.

3.1 Predstavitev pojmov teorije strežbe

Teorija strežbe povzeta po viru [19] je z vidika računalniških omrežij potrebna zaradi vse hitrejšega razvoja računalniških sistemov. Potrebujemo jo za lažje razumevanje, načrtovanje in merjenje zmogljivosti računalniških omrežij. Osnovni gradniki teorije strežbe so:

- *zahteve*: v računalniškem omrežju je to običajno paket, sestavljen iz podatkovnega in kontrolnega dela;
- *čakalna vrsta*: vmesni pomnilnik (angl. *buffer*), ki hrani zahteve;
- *strežnik*: element omrežja, katerega naloga je obdelava in pošiljanje zahtev;

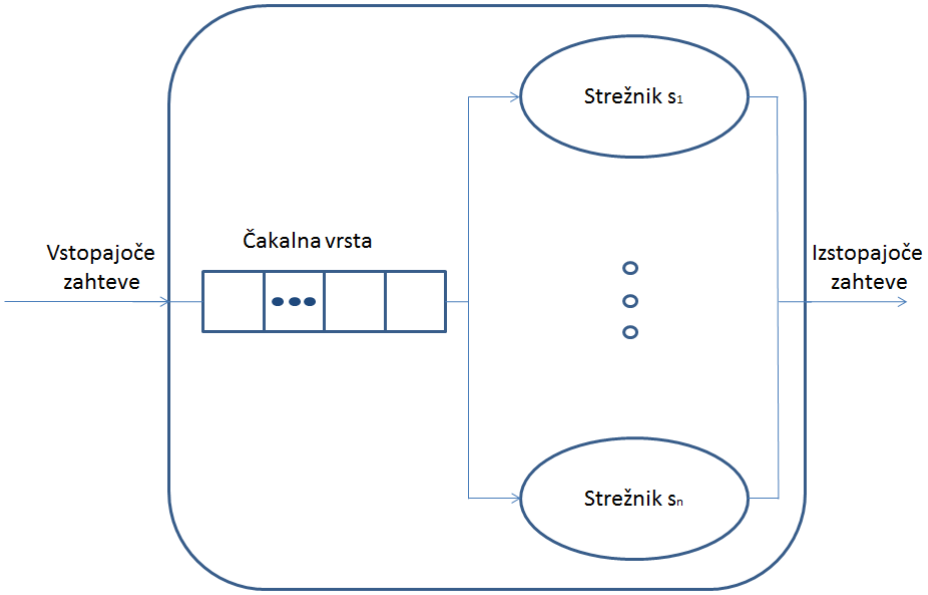


Slika 3.1 Model strežnega sistema.

Na sliki 3.1 je prikazan osnovni model strežnega sistema. Na levi strani vstopajo zahteve, ki predstavljajo vhodno breme sistema. V primeru, da je strežnik zaseden, gredo v čakalno vrsto, kjer počakajo, da se strežnik sprosti. Po končani strežbi zahteve model sistema zapustijo. Kompleksni strežni sistem lahko vsebuje več čakalnih vrst in strežnikov, ki so med seboj poljubno povezani. Čakalne vrste imajo dve pomembni značilnosti, in sicer velikost in disciplino. Velikost je lahko končna ali neskončna. V praksi obstajajo le končne vrste, saj neskončnih zaradi omejene velikosti pomnilnika ni mogoče realizirati. Neskončna čakalna vrsta se zaradi poenostavitve modela uporablja v matematični analizi. Strežna disciplina določa vrstni red jemanja zahtev iz čakalnih vrst. Poznamo naslednje vrste disciplin:

- FIFO (angl. *First In First Out*),
- LIFO (angl. *Last In First Out*),
- prioritetna disciplina,
- naključna disciplina,
- SJF (angl. *Shortest Job First*) in
- LJF (angl. *Largest Job First*).

Predhodno naštete discipline bomo podrobneje spoznali v naslednjem razdelku. Poznamo različne prioritete strežbe, ki jih ločimo na prekinitvene in neprekinitvene. Pri



Slika 3.2 Strežna enota.

prekinitvenem modelu prioritetnega procesiranja se v primeru, da vstopi zahteva z višjo prioriteto, kot jo ima tista v strežbi, strežba prekine in dodeli novo prispeli zahtevi. Pri neprekinitvenem modelu procesiranje se strežba zahteve nadaljuje do konca, nato pa se iz čakalne vrste vzame zahtevo z najvišjo prioriteto. Vsaki zahtevi je lahko tudi dodeljena le časovna rezina strežnika (angl. *time sharing*). V primeru, da zahteva ni dokončno postrežena, se vrne na konec čakalne vrste, in kasneje se ji dodeli nova časovna rezina. Poseben model strežnega sistema je *strežna enota*, prikazana na sliki 3.2. Strežna enota ima $n \geq 1$ paraleno vezanih strežnikov, ki so ekvivalentni. Zahteva bo postrežena natanko v enem izmed njih. Pred strežniki imamo eno čakalno vrsto. Poljubne vzporedne ali zaporedne vezave strežnih enot imenujemo *strežne mreže*.

Vsako strežno enoto lahko opišemo s Kendallovo notacijo tipa $A/B/m/k/P$. Pomen parametrov je naslednji:

- A: verjetnostna porazdelitev, ki predstavlja medprihodne čase vstopajočih zahtev,
- B: verjetnostna porazdelitev časa strežbe zahtev,
- m: število strežnikov, vezanih paralelno,

- k: kapaciteta strežne enote; dobimo jo kot vsoto števila strežnikov in dolžine čakalne vrste; pove nam koliko zahtev se lahko nahaja v strežni enoti in
- P: velikost populacije, ki v strežno enoto vstopa iz zunanjega okolja.

3.1.1 Osnovne značilnosti strežnih enot

V nadaljevanju dela bomo spoznali osnovne značilnosti strežnih enot, ki so potrebne za razumevanje kasnejšnih primerov.

Vhodni proces

Zanj je značilna intenzivnost porajanja zahtev, ki jo označujemo z λ. Definira nam medprihodne čase porajanja zahtev in se meri s povprečnim številom zahtev na časovno enoto.

Strežnik

Strežnik istočasno streže le eno zahtevo. Intenzivnost obdelave zahtev merimo s številom postreženih zahtev na časovno enoto μ.

Kapaciteta strežne enote

Kapaciteto strežne enote predstavlja število strežnikov in dolžina čakalne vrste. Pove nam največje število zahtev v strežni enoti.

Izhodni proces

Merimo ga z intenzivnostjo strežbe zahtev, ki je odvisna od intenzivnosti porajanja zahtev in intenzivnosti obdelave zahtev. Intenzivnost strežbe narašča s številom strežnikov, ki so vezani paralelno. V primeru enega strežnika je enaka intenzivnosti obdelave zahtev.

3.1.2 Zakon o ohranitvi pretoka

Za stabilne strežne enote velja, da je število vstopnih zahtev enako številu izhodnih zahtev gledano na daljši čas. V primeru, da je intenzivnost prihajanja zahtev prevelika, se čakalne vrste napolnijo in pride do nasičenja sistema, kar lahko vodi do izgubljanja zahtev.

3.2 Primera strežnih enot

V nadaljevanju bomo spoznali dva primera realnih izgubnih strežnih sistemov, kjer bomo predpostavili neskončno velikost populacije zahtev izven sistema.

3.2.1 Strežni sistem M/M/1/s

M/M/1/s predstavlja izgubni sistem s končno čakalno vrsto, ki je primeren za realno izvedbo. Neskončne čakalne vrste v praksi ni mogoče realizirati, zato se bomo osredotočili na sisteme s končnimi. Strežni sistem je sestavljen iz ene čakalne vrste in strežnika. V sistemu je lahko največ s zahtev od tega s-1 v čakalni vrsti in ena v strežbi. Zaradi končne kapacitete čakalne vrste gre za izgubni sistem. Verjetnost, da je sistem zasičen, je podana z izrazom

$$P_s = \frac{(1-\rho)\rho^s}{1-\rho^{s+1}}, \tag{3.1}$$

kjer ρ predstavlja faktor uporabnosti, ki je definiran z izrazom

$$\rho = \frac{\lambda}{\mu}. \tag{3.2}$$

Če je faktor uporabnosti manjši od 1 rečemo, da je sistem stabilen. V primeru, da je intezivnost porajanja zahtev enaka intenzivnosti obdelave zahtev, se nam izraz poenostavi v

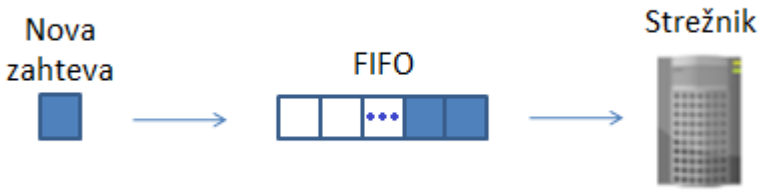
$$P_s = \frac{1}{s+1}. \tag{3.3}$$

Do zasičenosti pride, ko je v sistemu s zahtev. Pri vsaki novo prispeli zahtevi nastopi izgubljanje zahtev.

3.2.2 Strežni sistem M/M/m/m

M/M/m/m predstavlja strežni sistem brez čakalne vrste. Kapaciteta sistema je enaka številu strežnikov m. V primeru zasedenosti vseh strežnikov prihaja do izgub zahtev, zato gre tudi v tem primeru za izgubni sistem. Do izgub novo prispelih zahtev prihaja, ko je sistem nasičen in vsebuje m zahtev.

Za nas je zanimiv predvsem prvi primer, kjer se soočamo s končno čakalno vrsto. Potreben je mehanizem, ki se bo v primeru nasičenega sistema odločil, kaj storiti z novo prispelimi zahtevami. V nadaljevanju dela bomo zato spoznali algoritme, s katerimi rešujemo problem polnih čakalnih vrst.



Slika 3.3 Algoritem FIFO.

3.3 Algoritmi za razvrščanje paketov v čakalnih vrstah

V pričujočem razdelku bomo spoznali nekatere algoritme opisane v delih [20] in [21].

3.3.1 Algoritem FIFO

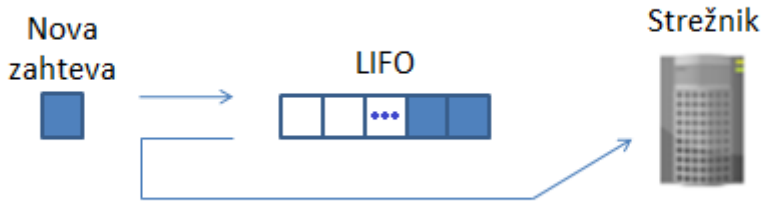
Algoritem FIFO (angl. *First In First Out*) prikazan na sliki 3.3 je najenostavnejši razvščevalnik zahtev. Vrstni red strežbe zahtev je enak vrstnemu redu prihoda zahtev v čakalno vrsto. Pri tem ne upošteva parametre kvalitete storitev, ampak zahteve samo prepošilja v strežbo glede na njihov čas prihoda. Takšen način razvrščanja slabo podpira kvaliteto storitev, saj mora novo prispela zahteva počakati vse tiste, ki so prispele v čakalno vrsto pred njo. Prednost takšnega algoritma je v zelo enostavni implementaciji in se še danes veliko uporablja v komunikacijskih omrežjih za zagotavljanje najboljših storitev (angl. *best effort*).

3.3.2 Algoritem LIFO

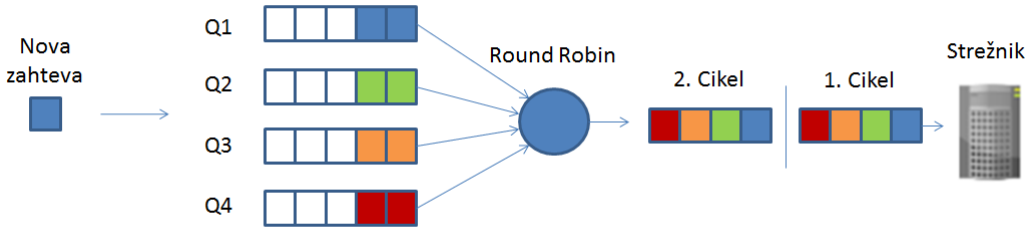
Za razliko od prejšnjega algoritma je jemanje zahtev iz čakalne vrste obratno. Algoritem LIFO (angl. *Last In First Out*) v strežbo pošilja zahtevo, ki je nazadnje prispela, kot je prikazano na sliki 3.4. Problem takšnega ravrščevalnika je v tem, da lahko pride do stradanja (angl. *starvation*) zahtev. To se zgodi v primeru, da zahtevo v čakalni vrsti ves čas prehitevajo novo prispele zahteve. Čas čakanja na strežbo je lahko zelo dolg, kar privede do stradanja zahteve.

3.3.3 Algoritem krožnega razvrščanja

Način delovanja algoritma krožnega razvrščanja (angl. *Round Robin*) je prikazan na sliki 3.5. Vsak tok zahtev ima svojo čakalno vrsto [21]. Razvrščevalnik pošilja zahteve v



Slika 3.4 Algoritem LIFO.

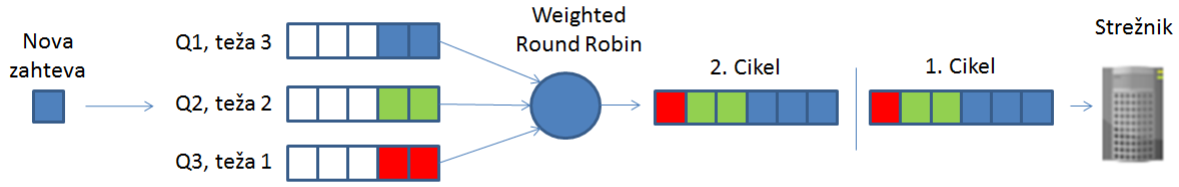


Slika 3.5 Algoritem krožnega razvrščanja, kjer Q označuje čakalno vrsto.

strežbo iz vsake čakalne vrste v krožnem zapredju. Primer jemanja zahtev iz čakalnih vrst bi bil torej sledeč Q1, Q2, Q3, Q4, Q1, Q2, ... Algoritem krožnega razvrščanja je enostaven za implementacijo. Zagotavlja pravičnost za vse vrste tokov in odlično izrabo pasovne širine. Problem se pojavi pri tokovih zahtev z višjo prioriteto, saj so obravnavani enako, kot vsi ostali in s tem ne pripomore pri zagotavljanju kakovosti storitve. Uporablja se predvsem v ATM omrežjih (angl. *Asynchronous Transfer Mode*), kjer omogoča doseganje visokih hitrosti s pošiljanjem paketov konstantne velikosti.

3.3.4 Algoritem obteženega krožnega razvrščanja

Algoritem obteženega krožnega razvrščanja (angl. *Weighted Round Robin*) se od predhodno opisanega krožnega razvrščanja razlikuje po tem, da so čakalne vrste obtežene. Teža



Slika 3.6 Algoritem obteženega krožnega razvrščanja.

predstavlja, koliko zahtev bo iz vsake čakalne vrste poslano v strežbo. S tem omogočimo boljši pretok nekaterim vrstam zahtev. Primer delovanja algoritma je prikazan na sliki 3.6. Implementiramo ga lahko na strojni način, ki je hitrejši in procesno manj zahteven od programskega načina. Pošiljanje zahtev v strežbo iz vsake čakalne vrste prepreči stradanje zahtev. Ko je neka čakalna vrsta prazna, algoritem razdeli pasovno širino med ostale čakalne vrste glede na njihovo obtežitev.

3.3.5 Naključni algoritem

Razvrščevalnik v primeru naključnega algoritma (angl. *random*) naključno izbere zahtevo iz vrste in jo pošlje v strežbo. Za izbor potrebuje generator naključnih števil. Ta nam pove zaporedno številko zahteve, ki bo poslana v strežbo. Generator naključnih števil mora biti hiter in računsko nepotraten. Običajno se pseudo naključno število izračuna po neki iterativni funkciji iz prejšnjega števila, zato moramo na začetku podati seme. Nad dobljenim številom nato izvedemo operacijo modul s številom zahtev v čakalni vrsti in dobimo številko zahteve, ki bo poslana v strežbo.

3.3.6 Algoritem SJF

Pri tem načinu razvrščanja algoritem SJF (angl. *Shortest Job First*) izbere zahtevo, ki ima najkraši čas obdelave in jo pošlje v strežbo.

3.3.7 Algoritem LJF

Za razliko od prejšnjega algoritma se tukaj izbere zahtevo z najdaljšim časom obdelave. Pri obeh algoritmih (SJF in LJF (angl. *Largest Job First*)) lahko nastopi že predhodno omenjeno stradanje zahtev.

3.3.8 Algoritem s prioritetami

Vsak tok zahtev ima svoj razred. Razred določa stopnjo kvalitete storitve in prioriteto. Zahteve se glede na njihove prioritete razvrstijo v različne čakalne vrste. Algoritem s prioritetami (angl. *Strict priority*) v strežbo pošlje zahtevo, ki se nahaja v čakalni vrsti z najvišjo prioriteto. Tukaj se srečamo s problemom stradanja zahtev. Nevarnost se pojavi pri zahtevah z nizko prioriteto, ki stradajo v primeru, ko jih ves čas prehitevajo zahteve z višjo prioriteto.

3.3.9 Algoritem obteženega krožnega razvrščanja s prioriteto

Algoritem obteženega krožnega razvrščanja s prioriteto (angl. *Weighted Round Robin with Priority Queuing*) [20] predstavlja kombinacijo prej opisanih algoritmov obteženega krožnega razvrščanja in razvrščanja s prioritetami. Razvrščevalnik najprej pošlje v strežbo zahteve, ki se nahajajo v čakalni vrsti z najvišjo prioriteto. V primeru, da je čakalna vrsta z najvišjo prioriteto prazna, oziroma ne zasede izhodne kapacitete, pošilja zahteve iz preostalih obteženih čakalnih vrst po principu algoritma obteženega krožnega razvrščanja. Takšen algoritem omogoča majhne zakasnitve, nihanje zakasnitev in izgube za zahteve z visoko prioriteto.

3.3.10 Pravičen algoritem z obteženim razvrščanjem

Pravičen algoritem z obteženim razvrščanjem (angl. *Weighted fair queueing*) za vsako prispelo zahtevo v čakalno vrsto izračuna virtualni končni čas strežbe. Pri izračunu upoštevamo velikost zahtev in obtežitev. Razvrščevalnik pošlje v strežbo zahevo z najkrajšim časom strežbe. Algoritem zagotavlja pravično delitev pasovne širine glede na obtežitev. Slabost algoritma je visoka računska zahtevost, ki je potrebna za izračun končnega časa strežbe.

3.3.11 Algoritem EDF

Pri algoritmu EDF (angl. *Earliest Deadline First*) [21] ima vsak tok zahtev določeno dopustno zakasnitev, ki jo bomo označili z d_i. Spremenljivka a_{ij} označuje čas prihoda zahteve j, ki pripada toku i. Končni rok (angl. *deadline*) zahteve j, dobimo kot vsoto časa prihoda zahteve in dopustne zakasnitvije toka, ki mu pripada. Matematični zapis bi izgledal takole: $a_{ij} + d_i$. Razvrščevalnik izbere zahtevo z najkrajšim končnim rokom in jo pošlje v strežbo. V splošnem bi lahko rekli, da gre za prioritetni algoritem, kjer prioriteto predstavlja čas. Zelo pogosto se uporablja v omrežjih, kjer natopajo multimedijske aplikacije, ki zahtevajo delovanje v realnem času.

3.4 Merila za oceno razvrščevalnih algoritmov

Z razvrščevalnimi algoritmi želimo doseči maksimalno izkoriščenost kapacitete sistema, pri tem pa zagotoviti pravičnost in kvaliteto storitve končnemu uporabniku. Za primerjavo algoritmov se uporabljajo naslednji parametri:

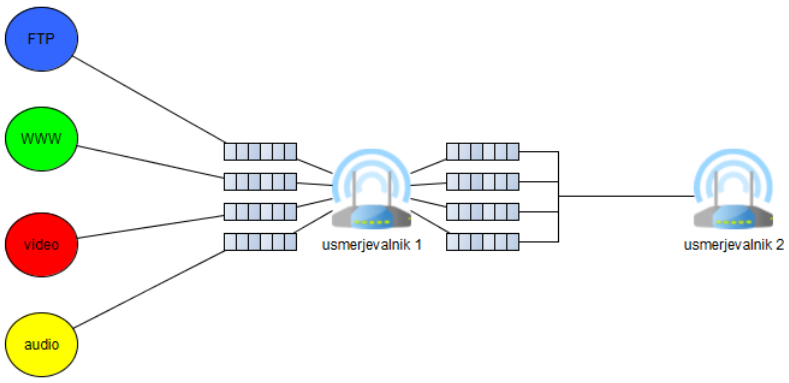
- Učinkovitost: Naloga algoritma je učinovito razvrščanje zahtev s katerimi zaposli omrežne vire in zadovolji končne uporabnike. Algoritem je učinkovit, če zagotavlja zmogljivost, ki omogoča kvaliteto storitev pri obremenjenih omrežjih.
- Prepustnost: Pove nam učinkovitost izrabe pasovne širine, ki je porazdeljena med različne tokove zahtev.
- Zaščita: Od algoritma želimo, da ima tok zahtev na preostale tokove zahtev čim manj vpliva. Med drugim želimo preprečiti stradanje zahtev in omogočiti vsem uporabnikom dostop do storitve, ki jo ponuja strežnik.
- Prilagodljivost: Meri nam podporo prilagajanja algoritma različnim zahtevam kvalitete storitve.
- Kompleksnost: Odvisna je od implementacije algoritma. Algoritem je lahko realiziran na programski ali strojni način. Pomembna je predvsem procesna zahtevnost, ki se z razvojem omrežja povečuje.
- Izgube paketov: Merimo jo v številu izgubljenih zahtev. Z razvrščevalnim algoritmom jih želimo zmanjšati.

- Latenca: Definira nam čas, ki preteče od izdaje zahteve s strani uporabnika do odziva sistema. Pri tem je pomembno tudi nihanje latence (angl. *jitter*).

4 Pregled obstoječih simulacij algoritmov razvrščanja

V pričujočem poglavju bomo spoznali tri primere obstoječih simulacij algoritmov razvrščanja. Ti so sledeči:

- Primer 1:
  - FIFO,
  - strežba po prioriteti in
  - strežba po prioriteti z možnostjo izpodrivanja paketov.
- Primer 2:
  - algoritem s prioriteto,
  - algoritem obteženega krožnega razvrščanja s prioriteto in
  - algoritem LLQ [20] (angl. *Low Latency Queuing*).
- Primer 3:
  - algoritem s prioriteto,



Slika 4.1 Shema omrežnih elementov primera 1.

- algoritem obteženega krožnega razvrščanja s prioriteto in
- algoritem LLQ (angl. *Low Latency Queuing*).

4.1 Primer 1

Prvi primer [7] predstavlja simulacijo delovanja 4 različnih aplikacij z dvema usmerjevalnikoma, kot je prikazano na sliki 4.1. Vsak omrežni element je imel štiri čakalne vrste, in sicer vhodno in izhodno čakalno vrsto za vsako smer prenosa. Pri simulaciji so se uporabile naslednje discipline strežbe:

- FIFO,
- strežba po prioriteti in
- strežba po prioriteti z možnostjo izpodrivanja paketov.

4.1.1 Breme

Breme je bilo predstavljeno s štirimi viri, ki imajo različne prioritete. Viri so bili sledeči:

- avdio: prenos zvoka,
- video: prenos video vsebine,
- www: brskanje po spletu,

Lastnost	Avdio	Video	WWW
Minimalna velikost paketa	30 B	640 B	10 B
Maksimalna velikost paketa	30 B	640 B	1400 B
Porazdelitev velikosti paketov	Konstantna	Konstantna	Enakomerna
Porazdelitev dolžine sporočil	Konstantna	Konstantna	Eksponentna

Tabela 4.1 Tabela prikazuje parametre aplikacij.

- ftp: prenos datotek.

Viri so razvrščeni v alinejah po prioriteti, kjer ima avdio najvišjo prioriteto, ftp pa najnižjo. Avdio in video predstavljata togi aplikaciji, preostali dve pa elastični aplikaciji.

4.1.2 Povezava

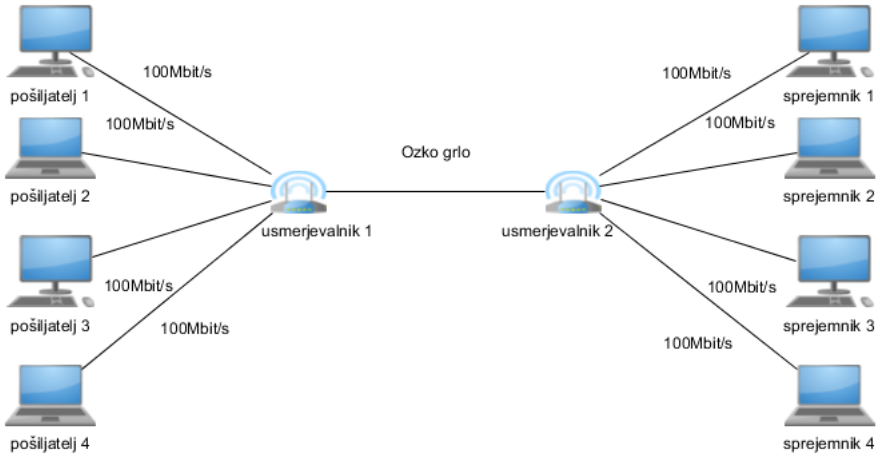
Hitrost povezave med usmerjavalnikoma je znašala 2048 kbps.

4.1.3 Simulacija

Aplikacije so imele definirane parametre, prikazane v tabeli 4.1. Pri ftp se je prenašala datoteka velikosti 100 kB. Simulacije so bile izvedene pri treh različnih obremenitvah, in sicer 50%, 90% in 100% obremenitvi povezave. Ob istih pogojih so bile testirane tri različne strežne discipline čakalnih vrst.

4.1.4 Rezultati

Rezultati simulacij so pokazali, da so pri FIFO disciplini na vhodni čakalni vrsti zakasnitve enakomerno razporejene med vse aplikacije. Izjema je bil ftp promet, ki je imel zaradi večje dolžine paketov nekoliko večjo zakasnitev. Strežna disciplina s prioriteto je pripomogla k zmanjšanju zakasnitve govornih paketov z 20 ms na 12 ms in video paketov pri 100% obremenjeni povezavi. Vzrok za zmanjšanje zakasnitve je upoštevanje prioritet, ker ima avdio in video promet višjo prioriteto kot ftp. Pri 50% obremenjeni povezavi so rezultati podobni kot pri FIFO strežni disciplini. Strežba po prioriteti z možnostjo izpodrivanja omogoča paketom z višjo prioriteto izpodrivati pakete z nižjo prioriteto iz čakalnih vrst. Rezultati simulacij so pokazali, da izpodrivanje paketov z nižjo prioriteto ne pripomore veliko k zmanjšanju zakasnitev prometa z višjo prioriteto v primerjavi s strežno disciplino po prioriteti. Ima pa vpliv na izgube paketov z nižjo prioriteto, ki



Slika 4.2 Na sliki je prikazana shema omrežja primera 2.

se v tem primeru povečajo. Pri vseh treh strežnih disciplinah lahko pri večji obremenitvi povezave opazimo povečanje zakasnitve zaradi daljših čakalnih vrst. Iz pridobljenih rezultatov simulacij lahko sklepamo, da se najbolje obnese algoritem s prioriteto, ki na račun prometa z nizko prioriteto zagotavlja boljšo kakovost storitev togim aplikacijam v obremenjenem omrežju.

4.2 Primer 2

Primer 2 [20] predstavlja simulacijo treh različnih algoritmov razvrščanja v omrežju prikazanem na sliki 4.2. Vsi omrežni elementi v vhodnih in izhodnih čakalnih vrstah uporabljajo razvrščevalni algoritem FIFO z izjemo usmerjavalnika 1, na katerem so bili uporabljeni naslednji trije razvrščevalni algoritmi:

- algoritem s prioriteto,
- algoritem obteženega krožnega razvrščanja s prioriteto in
- algoritem LLQ [20] (angl. *Low Latency Queuing*), ki predstavlja kombinacijo algoritma s prioriteto in pravičnega algoritma z obteženim razvrščanjem.

4.2.1 Breme

Breme predstavljajo trije tokovi, ki so jim dodeljeni različni razredi kakovosti storitve. Tok predstavlja zaporedje paketov z določeno pasovno širino.

	Tok 1	Tok 2	Tok 3
Teža	∞	8	3

Tabela 4.2 Tabela prikazuje parametre čakalne vrste.

Čas toka	1.5 - 16.5 s
Vrsta toka	UDP CBR
Velikost paketa	500 B
Interval paketov	5 ms
Pasovna širina	0.8 Mbit/s
Dolžina čakalne vrste	50 paketov

Tabela 4.3 Tabela prikazuje parametre tokov.

4.2.2 Povezava

Vse kapacitete povezav v omrežju so 100 Mbit/s z izjemo ozkega grla (glej sliko 4.2), katerega kapaciteta znaša od 0.1 Mbit/s do 3 Mbit/s s korakom 0.1 Mbit/s.

4.2.3 Simulacija

Parametri čakalne vrste so definirani v tabeli 4.2. Vsak tok ima definirane parametre, prikazane v tabeli 4.3. Ozko grlo ima začetno pasovno širino 1.2 Mbit/s, ki je nato naraščala s korakom 0.1 Mbit/s.

4.2.4 Rezultati

Rezultati simulacije so pokazali, da pri vseh treh razvrščevalnih algoritmih tok 1 dobi dodeljeno celotno pasovno širino, ki jo potrebuje. Ta znaša 0.8 Mbit/s. Algoritem obteženega krožnega razvrščanja s prioriteto in algoritem LLQ upoštevata striktno prioriteto pri toku z najvišjo prioriteto. Preostalih 0.4 Mbit/s se porazdeli med tokova 2 in 3 v razmerju 8 proti 3 glede na obtežitev. Tako znaša povprečna prepustnost za tok 2 0.291 Mbit/s in 0.109 Mbit/s za tok 3. Algoritem s prioriteto za razliko od prejšnjih dveh algoritmov dodeli preostalih 0.4 Mbit/s toku 2, ki ima višjo prioriteto od toka 3. V tem primeru pride do stradanja toka 3. S časom se čakalna vrsta napolni in prihaja do izgub paketov. Do stradanja zahtev prihaja pri vseh treh algoritmih, najbolj pa se to pozna pri algoritmu s prioriteto, ki toku 3 onemogoči strežbo. S povečanjem kapacitete ozkega

grla na 1.6 Mbit/s odpravimo izgube paketov toka 2. Z vsakim nadaljnim povečanjem kapacitete za 0.1 Mbit/s se zmanjšajo izgube paketov toka 3 za 13%, dokler ne dosežemo kapaciteto 2.4 Mbit/s, ki popolnoma odpravi izgube paketov. Glede na rezultate simulacije lahko sklepamo, da se bolje obneseta algoritem obteženega krožnega razvrščanja s prioriteto in algoritem LLQ. S tem ko zadovoljita tok z najvišjo prioriteto, omogočata pravičen dostop do preostale pasovne širine vsem tokovom glede na njihovo obtežitev in s tem zmanjšata izgube ter stradanje tokov z najnižjo prioriteto.

4.3 Primer 3

Primer 3 [20] predstavlja simulacijo omrežja prikazanega na sliki 4.2 pri različnih tokovih. Algoritmi so bili sledeči:

- algoritem s prioriteto,
- algoritem obteženega krožnega razvrščanja s prioriteto in
- algoritem LLQ (angl. *Low Latency Queuing*).

4.3.1 Breme

Breme predstavljajo štirje različni tipi prometa:

- VoIP,
- video,
- FTP in
- Web.

4.3.2 Povezava

Vse kapacitete povezav v omrežju so 100 Mbit/s z izjemo ozkega grla, katerega začetna kapaciteta je znašala 0.5 Mbit/s.

4.3.3 Simulacija

Paketi so bili razdeljeni v tri različne razrede kakovosti storitev. Glede na razred so se nato razvrstili v eno iz med treh čakalnih vrst. Parametri vrst so predstavljeni v tabeli 4.4, parametri prometa pa so prikazani v tabeli 4.5.

	Čakalna vrsta 1	Čakalna vrsta 2	Čakalna vrsta 3
Teža	∞	8	3
Dolžina čakalne vrste	20	20	20
Dodeljeni tokovi	VoIP	Video	FTP, Web

Tabela 4.4 Tabela prikazuje parametre čakalnih vrst.

	VoIP	Video	FTP	Web
Vrsta toka	RTP Real - Audio	UDP, CBR	TCP, FTP	UDP
Velikost paketov	210 B	1000 B	1040 B	500 B
Interval paketov	10 ms	30 ms	-	-
Pasovna širina	0.168 Mbit/s	0.5 Mbit/s	-	-
Čas toka	3 - 13 s	4 - 14 s	1 - 11 s	2 - 12 s

Tabela 4.5 Tabela prikazuje parametre prometa.

4.3.4 Rezultati

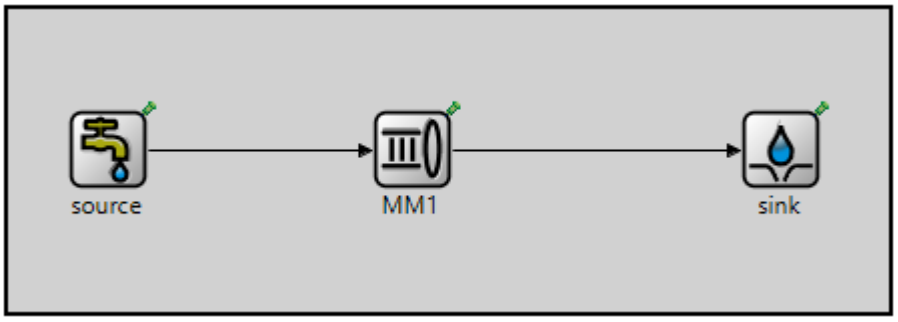
Rezultati simulacij so pokazali večjo zakasnitev video prometa pri algoritmu obteženega krožnega razvrščanja s prioriteto, kot pri ostalih dveh. Algoritem obteženega krožnega razvrščanja s prioriteto dodeli več pasovne širine FTP in Web prometu v primerjavi z LLQ in prioritetnim algoritmom, kar povzroči zakasnitve pri video prometu. Za video promet sta bolj primerna algoritem LLQ in prioritetni algoritem, kjer so bile zakasnitve video prometa manjše. Rezultati so pokazali tudi delovanje TCP mehanizma. Na začetku simulacije je FTP promet zasedal celotno pasovno širino. Ko se je pojavil preostali promet z višjo prioriteto so se pojavile izgube FTP paketov. TCP mehanizem je prilagodil hitrost pošiljanja FTP paketov in jo v tem primeru zmanjšal. S tem so se čakalne vrste spraznile in posledično zmanjšale zakasnitve.

5 Simulacije strežnega sistema M/M/1/s

V pričujočem poglavju bomo implementirali model strežnega sistema M/M/1/s s tremi različnimi algoritmi razvrščanja. Sledi izvedba simulacij, na podlagi katerih bomo algoritme med seboj primerjali. Na koncu bomo predstavili programsko okolje OMNeT++ in uporabljene module.

5.1 Predstavitev modelov za analizo razvrščevalnih algoritmov čakalnih vrst

Za simulacijo in analizo smo uporabili model strežnega sistema M/M/1/s, prikazan na sliki 5.1, implementiran v orodju OMNeT++ [22].



Slika 5.1 Model strežnega sistema M/M/1/s.

Model je sestavljen iz treh modulov, ki so izpeljani iz modula cSimpleModule, in sicer:

- *source*: generira zahteve tipa *cMessage*;
- *MM1*: vsebuje vhodno čakalno vrsto in omogoča strežbo; postreže prispelo zahtevo in jo pošlje naslednjemu modulu;
- *sink*: briše postrežene zahteve.

Vsak modul je sestavljen iz sledečih datotek:

- NED datoteka; opis parametrov modula; določitev modulov omrežja in povezav med njimi;
- C++ datoteka; zaglavna datoteka določa deklaracije spremenljivk in metod; izvorna datoteka implementira metode in funkcionalnost modula;
- INI datoteka; konfiguracijska datoteka, ki omogoča nastavljanje parametrov za izvajanje simulacije;

Moduli med seboj komunicirajo preko vhodnih in izhodnih vrat. Povezani so preko DatarateChannel, ki predstavlja prenosni kanal z zakasnitvijo 50 ns in pasovno širino 100 Mbit/s. Breme omrežja je bilo generirano z modulom *source*, ki je generiral zahteve tipa cMessage z 4 različnimi prioritetami. Hitrost pošiljanja zahtev je bila določena z eksponentno porazdelitvijo medprihodnih časov zahtev v datoteki *NED*. Generirane zahteve so bile poslane na izhodna vrata, ki so bile povezane z vhodnimi vrati modula *MM1*. Prispele zahteve so se shranjevala v vhodno čakalno vrsto, od koder je razvrščevalni algoritem določil, katera zahteva bo postrežena. Čas strežbe je bil določen z eksponentno

porazdelitev medprihodnih časov zahtev	eksponentna
porazdelitev časa strežbe	eksponentna
kapaciteta vhodne čakalne vrste	100 paketov
število generiranih paketov	200 paketov
pasovna širina	100 Mbit/s
zakasnitev povezave	50 ns

Tabela 5.1 Konfiguracija strežnega sistema M/M/1/s.

porazdelitvijo. Vhodna čakalna vrsta strežnega sistema *MM1* je imela implementirane tri različne algoritme razvrščanja, in sicer:

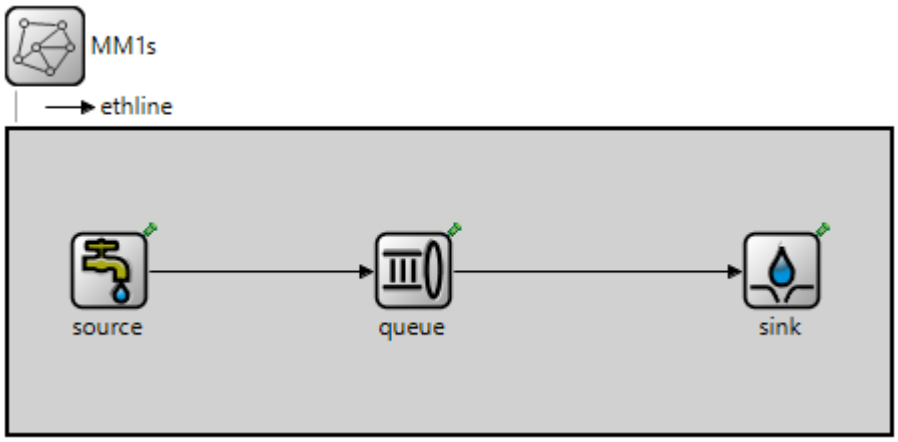
- FIFO (angl. *First In First Out*),
- algoritem obteženega krožnega razvrščanja (angl. *Weighted Round Robin, WRR*) in
- algoritem s prioriteto (angl. *Strict priority, SP*).

Postrežene zahteve so bile nato poslane modulu *sink*, ki je izračunal njihovo latenco in jih izbrisal. Latenca predstavlja čas potovanja zahteve od izvora do ponora. Sestavljena je iz časa potovanja zahteve, časa čakanja v vhodni čakalni vrsti in časa strežbe. Rezultate simulacij smo beležili z uporabo signalov, ki omogočajo modulom oddajanje vrednosti med potekom simulacije. Pridobljene podatke smo shranjevali v izhodne datoteke skalarjev in vektorjev. Za vse simulacije smo uporabili nastavitve parametrov, prikazane v tabeli 5.1

5.2 FIFO analiza M/M/1/s sistema

Sledi opis izvedbe simulacij in analiza rezultatov simulacij FIFO razvrščevalnega algoritma.

5.2.1 Simulacija



Slika 5.2 Strežni sistem M/M/1/s s FIFO strežno disciplino.

Na sliki 5.2 je prikazan model strežnega sistema M/M/1/s, kjer modul *queue* vsebuje FIFO strežno disciplino. Modul *source* generira zahteve s 4 različnimi prioritetami, kjer prioriteta 1 predstavlja najnižjo prioriteto, prioriteta 4 pa najvišjo. Za vsako prioriteto je bilo v času simulacije 107 sekund generiranih 50 zahtev, kar skupaj znese 200 zahtev. Zahteve so bile nato poslane modulu *queue*, ki poleg strežbe vsebuje tudi vhodno čakalno vrsto velikosti 100 paketov. Vrstni red jemanja zahtev iz vhodne čakalne vrste deluje po principu FIFO in je enak vrstnemu redu prihajanja zahtev. Postrežene zahteve so bile poslane modulu *sink*, ki jih nato zavrže. Za izvedbo simulacije smo uporabili konfiguracijo, prikazano v tabeli 5.1. Vsako simulacijo smo ponovili trikrat in nato vzeli povprečni izid. S tem smo zmanjšali verjetnost odstopanja od meritev. Primer meritve povprečnega časa čakanja v vhodni čakalni vrsti za zahteve z različnimi prioritetami je prikazana v tabeli 5.2. V nadaljevanju so predstavljeni grafi, kjer so prikazani povprečni rezultati simulacij.

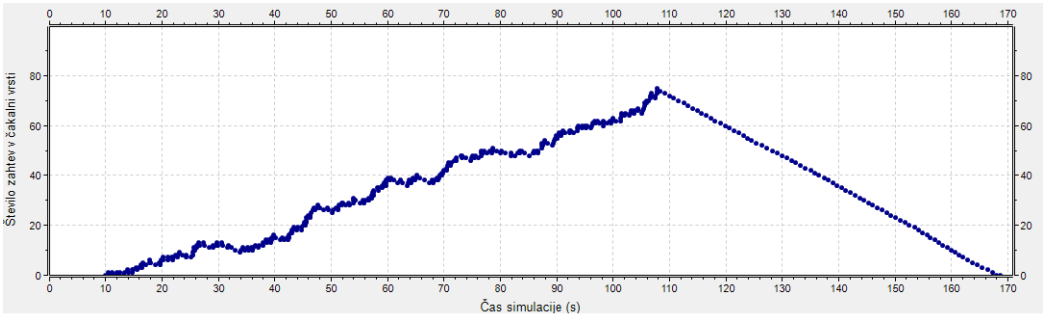
5.2.2 Rezultati

Grafa 5.3 in 5.4 prikazujeta spreminjanje zasedenosti čakalne vrste in čakalnih časov zahtev v vhodni čakalni vrsti med izvajanjem simulacije. Rezultati simulacij so pokazali, da se čas čakanja zahtev povečuje s številom zahtev, ki se nahajajo v čakalni vrsti. Novo prispela zahteva mora počakati, da so postrežene zahteve, ki se že nahajajo v čakalni

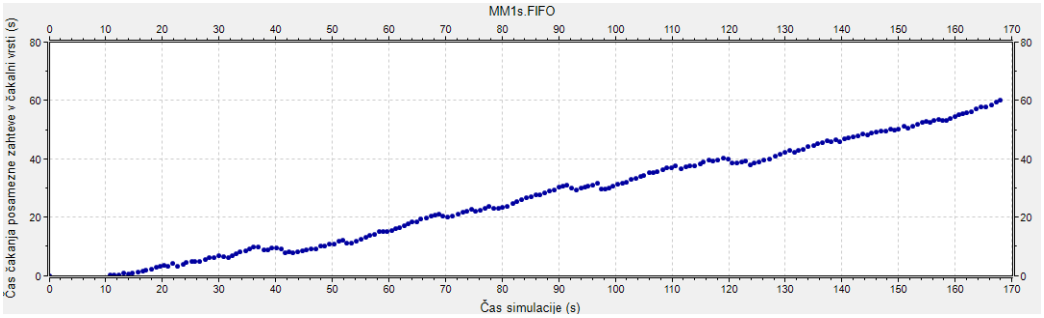
	1. ponovitev	2. ponovitev	3. ponovitev
zahteve s prioriteto 1	25.325 s	27.083 s	27.462 s
zahteve s prioriteto 2	26.198 s	28.019 s	28.376 s
zahteve s prioriteto 3	26.092 s	27.829 s	28.212 s
zahteve s prioriteto 4	25.642 s	27.436 s	27.813 s

Tabela 5.2 Ponovitev meritev povprečnih čakalnih časov v vhodnih čakalnih vrstah glede na prioritete zahtev za strežni sistem M/M/1/s s FIFO strežno disciplino.

vrsti. Največjo zasedenost čakalne vrste smo dosegli pri času 107 s, in sicer 75 zahtev, ko so bile prejete vse generirane zahteve. Sledila je strežba zahtev, ki so se nahajale v čakalni vrsti in s tem tudi praznjenje čakalne vrste.



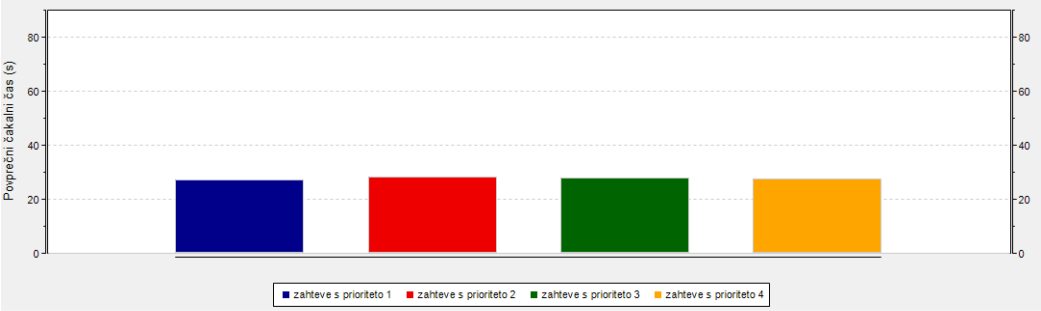
Slika 5.3 Spreminjanje števila zahtev v FIFO vhodni čakalni vrsti.



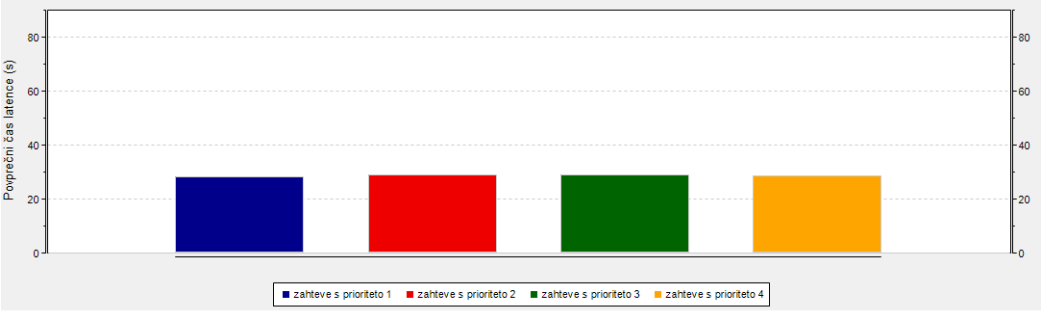
Slika 5.4 Čakalni čas posamezne zahteve v FIFO vhodni čakalni vrsti.

Graf 5.5 prikazuje povprečni čakalni čas zahtev v čakalni vrsti glede na prioriteto

zahtev. Opazimo, da je čakalni čas za vse prioritete zahtev približno enak, saj FIFO strežna disciplina ne upošteva prioritet in vse zahteve obravnava enako. Graf 5.6 prikazuje povprečno latenco zahtev. Iz grafov 5.5 in 5.6 lahko opazimo, da največ časa prispeva čakalni čas v čakalni vrsti in sicer 97% časa latence.



Slika 5.5 Povprečni čakalni čas zahtev v vhodni čakalni vrsti.



Slika 5.6 Povprečna latenca zahtev.

5.3 SP analiza M/M/1/s sistema

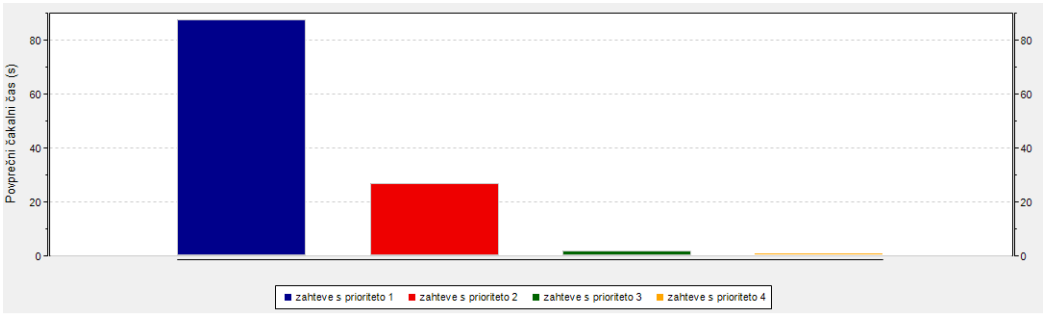
Sledi opis izvedbe simulacij in analiza rezultatov simulacij SP razvrščevalnega algoritma.

5.3.1 Simulacija

Slika 5.2 prikazuje model strežnega sistema M/M/1/s s prioritetno strežno disciplino. Modul *source* generira zahteve s 4 različnimi prioritetami. Modul *queue* vsebuje algoritem s prioriteto, ki ima poleg vhodne čakalne vrste še 4 čakalne vrste z različnimi prioritetami. Razvrščevalni algoritem zahtevo iz vhodne čakalne vrste razvrsti v eno iz med 4 prioritetnih čakalnih vrst glede na prioriteto zahtev. Primer: zahtevi z najvišjo prioriteto se dodeli čakalna vrsta z najvišjo prioriteto. Vrstni red jemanja zahtev je določen s prioriteto. Najprej so potrežene zahteve iz čakalne vrste z najvišjo prioriteto, nazadnje pa tiste iz čakalne vrste z najnižjo prioriteto. Zahteve znotraj iste čakalne vrste imajo enako prioriteto in so postrežene po principu FIFO.

5.3.2 Rezultati

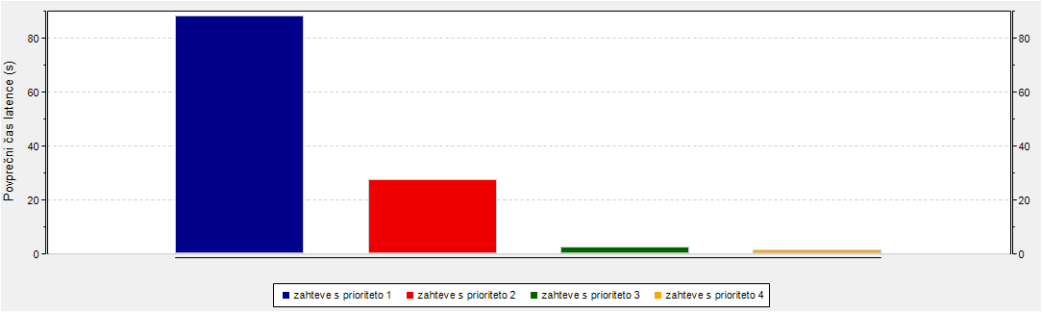
Na grafu 5.7 je prikazan povprečni čakalni čas zahtev v čakalnih vrstah. Zahtevem z visoko prioriteto 3 in 4 se je čas čakanja zmanjšal na račun zahtev z najnižjo prioriteto. S tem smo zagotovili boljšo kakovost storitve zahtevam z višjo prioriteto. Opazimo stradanje zahtev z najnižjo prioriteto, kjer se je čas čakanja trikrat povečal v primerjavi s FIFO strežno disciplino. Na grafu 5.8 je prikazana povprečna latenca zahtev. Pri zahtevah z najvišjo prioriteto čas čakanja predstavlja le še 35% časa latence, med tem ko pri zahtevah z najnižjo prioriteto znaša 99%, kar je razvidno iz grafov 5.7 in 5.8.



Slika 5.7 Povprečen čakalni čas zahtev v čakalnih vrstah.

	teža
čakalna vrsta s prioriteto 4	4
čakalna vrsta s prioriteto 3	3
čakalna vrsta s prioriteto 2	2
čakalna vrsta s prioriteto 1	1

Tabela 5.3 Obtežitev čakalnih vrst.



Slika 5.8 Povprečna latenca zahtev.

5.4 WRR analiza M/M/1/s sistema

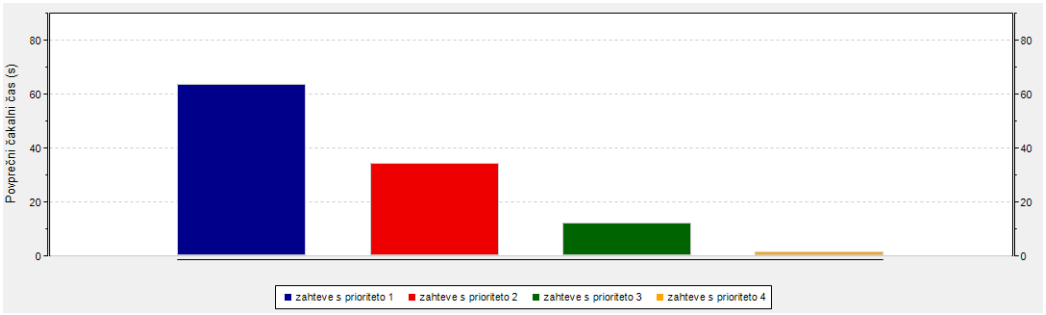
Sledi opis izvedbe simulacij in analiza rezultatov simulacij WRR razvrščevalnega algoritma.

5.4.1 Simulacija

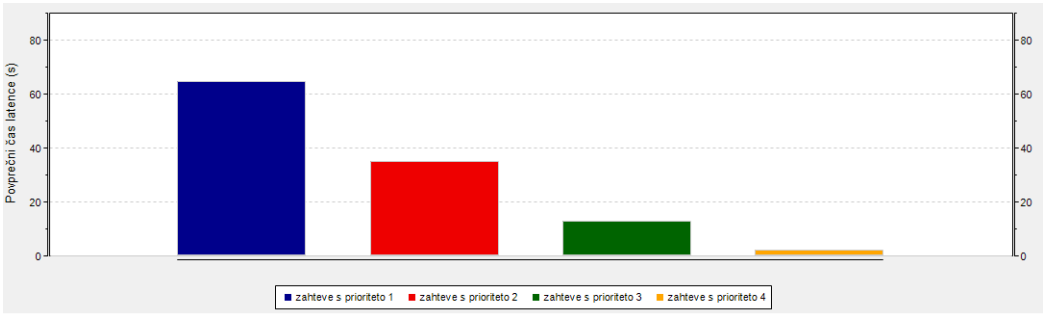
Slika 5.2 prikazuje model strežnega sistema M/M/1/s z obteženim krožnim razvščanjem. Modul *source* generira zahteve s 4 različnimi prioritetami. Modul *WRRQueue* vsebuje poleg vhodne čakalne vrste še 4 čakalne vrste, ki so različno obtežene. Obtežitev čakalnih vrst je prikazana v tabeli 5.3. Razvščevalni algoritem zahtevo iz vhodne čakalne vrste razvrsti v eno iz med 4 čakalnih vrst glede na prioriteto zahtev. Znotraj iste čakalne vrste se nahajajo zahteve z enako prioriteto. Razvrščevalnik nato v vsakem ciklu obišče posamezno čakalno vrsto, kjer v strežbo pošlje število zahtev določeno z utežjo čakalne vrste.

5.4.2 Rezultati

Na grafih 5.9 in 5.10 so prikazani rezultati povprečne latence in povprečnih čakalnih časov zahtev v čakalni vrsti. Opazimo zmanjšanje časa čakanja in latence pri zahtevah z višjo prioriteto, kar pripomore k boljšemu zagotavljanju kakovosti storitve. Čas čakanja zahtev z nižjo prioriteto se zato poveča, vendar s krožnim razvrščanjem preprečimo stradanje zahtev z nizko prioriteto za razliko od prioritetnega razvrščanja, kot je prikazano na grafih 5.9 in 5.7.



Slika 5.9 Povprečen čakalni čas zahtev v čakalni vrsti.



Slika 5.10 Povprečna latenca zahtev.

5.5 Povzetek rezultatov simulacij

Rezultati simulacij so pokazali, da čas čakanja v čakalnih vrstah močno vpliva na zakasnitev zahtev. Najbolje se je obnesel algoritem obteženega krožnega razvrščanja, ki je zagotovil boljšo kakovost storitve zahtevam z višjo prioriteto in hkrati preprečil stradanja zahtev z nizko prioriteto. Algoritem bi bil primeren za toge aplikacije kot so prenos

zvoka in videa v realnem času, ki so občutljive na zakasnitve. Pri FIFO strežni disciplini opazimo, da ne pripomore k izboljšanju kakovosti storitve, saj ne upošteva prioritete prometa. Takšen algoritem je enostavno implementirati in se uporablja v omrežjih, kjer nastopajo elastične aplikacije, ki niso občutljive na zakasnitve. Težave se pojavijo pri prometu, ki potrebuje večji delež omrežnih virov za boljšo kakovost storitve, kar pa FIFO strežna disciplina ne omogoča. To nam omogoča algoritem s prioritetnim razvrščanjem, ki je zagotovil najboljšo kakovost storitev zahtev z visoko prioriteto, vendar na račun stradanja zahtev z najnižjo prioriteto. Algoritem bi bil primeren za manj obremenjena omrežja, kjer bi bila prepustnost dovolj velika, da bi preprečili stradanje zahtev z najnižjo prioriteto.

5.6 Orodje OMNeT++

Za izvedbo simulacij smo uporabili orodje OMNeT++ [19], ki je sestavljeno iz knjižnic implementiranih v programskem jeziku C++. Orodje nam omogoča postavitev simulacijskega modela, izvedbo simulacije, zbiranje in analizo rezultatov simulacije. Vsebuje komponente napisane v programskem jeziku C++, ki se povezujejo v višje module in komponente modela z uporabo jezika NED (angl. *NEtwork Description*). Konfiguracijo nastavitev za izvajanje simulacije določimo v datoteki s končnico *ini*. Poznamo enostavne in sestavljene module, ki so lahko sestavljeni iz enega ali več enostavih ali ostalih sestavljenih modulov. Funkcionalnost modulov določa programska koda C++. Moduli med seboj komunicirajo s pošiljanjem sporočil preko vrat in povezav. Pri eksperimentu smo uporabili naslednje module [23]:

- cSimpleModule,
- cMessage,
- cDatarateChannel,
- cGate,
- cQueue.

5.6.1 cSimpleModule

Modul cSimpleModule predstavlja podrazred, ki je izpeljan iz razreda cModule. Iz razreda cSimpleModule izhajajo enostavni moduli. Modul ima deklarirane sledeče metode:

- initialize(); omogoča inicializacijo; kliče se po kreiranju modula pred prvim dogodkom;
- handleMessage(cMessage *msg); klic ob vsakem sprejemu sporočila; določa odziv na prejeto sporočilo;
- finish(); klic ob koncu simulacije; metoda primerna za obdelavo podatkov pridobljenih iz simulacije;

5.6.2 cMessage

Objekt cMessage predstavlja dogodek oziroma sporočilo, ki se prenaša med moduli. Nastavimo mu lahko sledeče atribute: tip sporočila, prioriteta in časovni žig. Vsako sporočilo hrani informacije, kdaj je bilo poslano, čas prejema, čas ko je bilo sporočilo nazadnje poslano, številko prejemnih vrat in modula. Za pošiljanje in brisanje sporočil so na voljo sledeče metode:

- send(cMessage *msg, cGate *outputgate); pošlje sporočilo na izhodna vrata;
- sendDelayed (cMessage *msg, simtime_t delay, int gateid); pošiljanje sporočil z zakasnitvijo;
- scheduleAt(simtime_t t, cMessage *msg); po času t modul pošlje sporočilo samemu sebi;
- cancelEvent (cMessage *msg); preklic dogodka, ki je bil poslan samemu sebi;
- cancelAndDelete(cMessage *msg); omogoča preklic in brisanje dogodka.

5.6.3 cDatarateChannel

cDatarateChannel predstavlja prenosni kanal med različnimi moduli. Določimo lahko zakasnitve, verjetnost napak in pasovno širino prenosnega kanala.

5.6.4 cGate

cGate predstavlja vrata, ki so običajno med seboj povezana preko povezav. Poznamo tri vrste vrat, in sicer vhodna, izhodna in dvosmerna vrata.

5.6.5 cQueue

Razred cQueue predstavlja čakalno vrsto, kamor se shranjujejo zahteve. Privzeto deluje po principu FIFO, kjer z metodo insert() vstavljamo zahteve v čakalno vrsto in metodo pop(), s katero jemljemo zahteve iz čakalne vrste. Lahko ji določimo, da deluje kot prioritetna čakalna vrsta. V tem primeru je potrebno implemetirati funkcijo za primerjanje prioritet, ki jo podamo v konstruktorju ob kreiranju čakalne vrste. Čakalni vrsti lahko določimo kapaciteto. Na voljo imamo še nekatere preostale metode, kot so:

- clear(); pobriše sporočila v čakalni vrsti;
- isEmpty(); uporablja se za preverjanje praznosti čakalne vrste;
- length(); pove nam dolžino čakalne vrste.

Vsa programska koda je priložena originalu diplomskega dela v originalni obliki.

6 Zaključek

V diplomskem delu smo spoznali delovanje algoritmov, ki skrbijo za razvrščanje zahtev v čakalnih vrstah. Izbrane algoritme smo nato tudi implementirali in izvedli simulacije za njihovo primerjavo. Preden smo se tega lotili, smo se spoznali z osnovnimi pojmi teorije strežbe, kjer imajo pomembno vlogo vhodne čakalne vrste, ki hranijo pakete preden gredo v strežbo. Za običajen promet v omrežju, kjer nastopajo elastične aplikacije, nam zadostuje najpreprostejša FIFO strežna disciplina. Toge aplikacije, ki delujejo v realnem času, (primer je prenos zvoka in videa), so zelo občutljive na zakasnitve. Zakasnitve so sestavljene iz fiksnega in spremenljivega dela. Na fiksni del z razvrščevalnimi algoritmi ne moremo vplivati, saj je odvisen od hitrosti in dolžine fizičnih povezav med pošiljateljem in prejemnikom. Lahko pa vplivamo na spremenljivi del zakasnitve, ki je odvisen od čakalnih časov zahtev v čakalnih vrstah na poti med pošiljateljem in prejemnikom. Potrebni so naprednejši algoritmi, ki nam omogočajo boljši pretok prometa z višjo prioriteto in s tem pripomorojo k izboljšanju kakovosti storitve. Z vidika toerije strežbe smo predstavili dva primera realnih strežnih sistemov M/M/1/s in M/M/m/m. Zaradi končne čakalne vrste gre za izgubna sistema. Spoznali smo se z različnimi razvrščevalnimi

algoritmi. Za simulacijo smo izbrali M/M/1/s strežni izgubni sistem. Preden smo se lotili eksperimenta, smo preučili 3 obstoječe izvedbe simulacij razvrščevalnih algoritmov. Za primerjavo smo izbrali tri algoritme, in sicer FIFO algoritem, algoritem s prioriteto in algoritem z obteženim krožnim razvrščanjem. Naštete algoritme smo implemetirali v programskem orodju OMNeT++, ki nam poleg implementacije omogoča tudi izvedbo simulacije in analizo rezultatov. Rezultati simulacij so pokazali, da ima čas čakanja v čakalnih vrstah velik vpliv na zakasnitve. Najbolje se je obnesel algoritem z obteženim krožnim razvrščanjem, ki je poleg izboljšanja kakovosti storitev zahtev z višjo prioriteto, omogočil tudi prepustnost zahtevam nižje prioritete. FIFO algoritem je obravnaval ves promet enako in je primeren za običajen promet, kjer ni potrebe po visoki stopnji kakovosti storitve. Za časovno kritične aplikacije, ki delujejo v realnem času, bi se najbolje obnesel algoritem s prioriteto, ki zagotavlja visoko stopnjo kakovosti storitve zahtev z visoko prioriteto na račun zahtev z nizko prioriteto. Slabost takšnega algoritma je stradanje prometa z nizko prioriteto. Tukaj se srečamo z vprašanjem ali so pridobitve zahtev z višjo prioriteto večje, kot pa izgube na račun zahtev z nižjo prioriteto. Takšen pristop je nujno potreben pri elastičnih aplikacijah, ki potrebujejo višjo stopnjo kakovosti storitve, pri tem pa želimo čim manjše izgube togih aplikacij, ki bi bile še sprejemljive za končnega uporabnika. Zavedati se moramo, da s simulacijo popolnih realnih razmer žal ne moremo doseči. Nadaljne raziskave bi lahko bile narejene v smeri nadgradnje obstoječih algoritmov ter izvedba meritev v realnem omrežju.

Literatura

[1] Science Dissemination Unit;. Dostopal: 06-02-2015. http://sdu.ictp.it/pinger/pinger.html.

[2] Iztok Humar, Mitja Golja, Jane Bešter. Mehanizmi zagotavljanja QoS pri pretočnem videu v omrežjih IP;. Dostopal: 04-02-2015. http://www.ltfe.org/wp-content/pdf/Mehanizmi_zagotavljanja_QoS.pdf.

[3] Lončarič T, Vehovec A, Kastelic M, Drofenik D, Divjak S, Kavčič A, et al.. e-računalništvo;. Dostopal: 07-01-2015. http://www.s-sers.mb.edus.si/gradiva/rac/moduli/podatkovne_baze/08_aplikacije/01_datoteka.html.

[4] OpenWrt Wiki. Linux Packet Scheduling;. Dostopal: 08-01-2015. http://wiki.openwrt.org/doc/howto/packet.scheduler/packet.scheduler.theory.

[5] Slabosti protokola TCP/IP;. Dostopal: 15-01-2015. http://wiki.fmf.uni-lj.si/wiki/%C5%A0tevilka_IP.

[6] Kos A, Verlic R, Tomazic S. Kakovost storitve v paketnih omrezjih. Elektrotehniški vestnik. 2004;71(3):103–108.

[7] Borut Klepec, Anton Kos. Vpliv diferenciacije IP prometa na delovanje aplikacij v realnem času;. Dostopal: 02-02-2015. http://www.ltfe.org/wp-content/pdf/Dif_IP_prometa.pdf.

[8] Raj Jain. Quality of Service Over IP;. Dostopal: 06-02-2015. http://www.cse.wustl.edu/~jain/talks/ftp/ipqos.pdf.

[9] Peter R Egli. INTRODUCTION TO QUALITY OF SERVICE CONCEPTS AND PROTOCOLS;. Dostopal: 07-02-2015. http://www.indigoo.com/dox/itdp/13_Management&Backbone/QoS.pdf.

[10] Resource Reservation Protocol. Wikipedia;. Dostopal: 16-01-2015. http://en.wikipedia.org/wiki/Resource_Reservation_Protocol.

[11] Braden B, Zhang L, Berson S, Herzog S, Jamin S. RFC 2205; 1997. Dostopal: 09-02-2015. https://tools.ietf.org/html/rfc2205.

[12] Server (computing). Wikipedia;. Dostopal: 10-01-2015. http://en.wikipedia.org/wiki/Server_%28computing%29.

[13] 1995 - 2014 Standard Performance Evaluation Corporation. Server Efficiency Rating Tool;. Dostopal: 14-01-2015. https://www.spec.org/sert/.

[14] Standard Performance Evaluation Corporation (SPEC). Server Efficiency Rating Tool (SERT) User Guide 1.0.2;. Dostopal: 14-01-2015. https://www.spec.org/sert/docs/userguide.pdf.

[15] Standard Performance Evaluation Corporation (SPEC). SPECpower Committee, Server Efficiency Rating Tool Draft Design Document;. Dostopal: 14-01-2015. http://www.energystar.gov/sites/default/files/specs//SERT_DraftDesignDocument.pdf.

[16] partyinfo; 2013. Dostopal: 12-02-2015. http://partyinfo.si/novice/naval-na-vstopnice-za-ultra-europezrusil-se-je-steznik.

[17] Wood, Anthony and Stankovic, John A. Denial of service in sensor networks. Computer. 2002;35(10):54–62.

[18] Računalniške novice; 2011. Dostopal: 10-02-2015. http://www.racunalniske-novice.com/novice/dogodki-in-obvestila/dos-napadi-na-wordpress.html.

[19] Miha Mraz in Miha Moškon. Modeliranje računalniških omrežij; 2015. Dostopal: 04-02-2015. http://lrss.fri.uni-lj.si/sl/teaching/mro/lectures/mro_2014.pdf.

[20] Tomáš Balogh, Martin Medvecký. Comparison of Priority Queuing Based Scheduling Algorithms. Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava; April 2010. Dostopal: 07-02-2015. http://webcache.googleusercontent.

com/search?q=cache:-7vNCqFO9GMJ:www.elektrorevue.cz/en/download/comparison-of-priority-queuing-based-scheduling-algorithms/+&cd=2&hl=en&ct=clnk&gl=si.

[21] Tsung-Yu-Tsai, Yao-Liang Chung, Zsehong Tsai. Introduction to Packet Scheduling Algorithms for Communication Networks. Institute for Information Industry, Graduate Institute of Communication Engineering, National Taiwan Univesity, Taipei, Taiwan, R.O.C;. Dostopal: 08-02-2015. http://www.intechopen.com/books/communications-and-networking/introduction-to-packet-scheduling-algorithms-for-communication-networks-.

[22] Modeliranje računalniških omrežij, 2. laboratorijska vaja, zgled modela sistema M/M/1; 2013/2014. Dostopal: 15-02-2015. https://ucilnica1314.fri.uni-lj.si/course/view.php?id=209.

[23] OMNeT++ API Reference 4.6;. Dostopal: 06-02-2015. http://www.omnetpp.org/doc/omnetpp/api/main.html.